

IN THE CLAIMS:

Please rewrite the claims to read as follows:

1 I, (Currently Amended) A method for executing uniprocessor (UP) coded workloads in a
2 multiprocessor (MP) computer system without having to rewrite the UP-coded work-
3 loads' code, comprising:

4
5 identifying threads in the uniprocessor coded workloads (UP-workloads) which
6 can execute concurrently, and identifying threads in the UP-workloads which cannot exe-
7 cute concurrently;

8
9 _____ assigning first threads which cannot execute concurrently to a first concurrency
10 group, and assigning second threads which cannot execute concurrently to a second con-
11 currency group, where any thread in the first concurrency group can execute concurrently
12 with any thread in the second concurrency group;

13
14 organizing the UP-coded workloads into one or more concurrency groups,
15 wherein UP-coded workloads in the same concurrency group are not permitted to execute
16 concurrently with one another in the MP computer system;

17
18
19 compiling the UP workload to execute on the MP system, the threads in the first
20 concurrency group compiled to be prevented from executing concurrently, the threads in
21 the second concurrency group compiled to be prevented from executing concurrently, and
22 the threads in the first and the second concurrency groups compiled to execute concur-
23 rently;

25 scheduling first and second execution vehicles that respectively execute on differ-
26 ent processors in the MP computer system at substantially the same time;

27

28 acquiring the ~~[[a]]~~ first concurrency group by the first execution vehicle and
29 the ~~[[a]]~~ second concurrency group by the second execution vehicle; and

30

31 executing UP-coded workloads in the first concurrency group through the first
32 execution vehicle at substantially the same time as UP-coded workloads in the second
33 concurrency group are executed through the second execution vehicle.

1 2. (Original) The method according to claim 1, wherein the UP-coded workloads are UP-
2 coded threads, and the first and second execution vehicles are first and second processes.

1 3. (Original) The method according to claim 1, wherein the UP-coded workloads are
2 messages, and the first and second execution vehicles are first and second threads.

1 4. (Original) The method according to claim 1, wherein the step of acquiring the first and
2 second concurrency groups further comprises:

3 dequeuing from a concurrency-group run queue a first concurrency-group data
4 structure associated with the first concurrency group; and

5 dequeuing from the concurrency-group run queue a second concurrency-group
6 data structure associated with the second concurrency group.

1 5. (Original) The method according to claim 4, further comprising:

2 setting a first CG flag in the first concurrency-group data structure to a value indi-
3 cating that the first concurrency group is in a running state; and

4 setting a second CG flag in the second concurrency-group data structure to a
5 value indicating that the second concurrency group is in a running state.

- 1 6. (Original) The method according to claim 4, further comprising:
2 appending UP-coded workloads enqueued on a first current queue in the first con-
3 currency-group data structure onto a first active queue in the first concurrency-group data
4 structure; and
5 appending UP-coded workloads enqueued on a second current queue in the sec-
6 ond concurrency-group data structure onto a second active queue in the second concur-
7 rency-group data structure.
- 1 7. (Original) The method according to claim 6, further comprising:
2 dequeuing UP-coded workloads in the first and second concurrency groups from
3 the first and second active queues, respectively; and
4 executing the dequeued UP-coded workloads to completion.
- 1 8. (Original) The method according to claim 5, further comprising:
2 in response to the first execution vehicle finishing execution of the UP-coded
3 workloads in the first concurrency group, the first execution vehicle performing the steps:
4 A) if at least one UP-coded workload in the first concurrency group is
5 executable;
6 (i) setting the value of the first CG flag to a value indicat-
7 ing that the first concurrency group is in a queued state;
8 (ii) re-enqueuing the first concurrency-group data struc-
9 ture onto the concurrency-group run queue;
10 B) if there are not any UP-coded workloads in the first concurrency
11 group that are executable, setting the first CG flag to a value indicating that the
12 first concurrency group is in a suspended state;
13 C) dequeuing from the concurrency-group run queue a third concur-
14 rency-group data structure associated with a third concurrency group; and
15 D) setting a third CG flag in the third concurrency-group data structure to
16 a value indicating that the third concurrency group is in a running state.

1 9. (Original) The method according to claim 1, wherein at least one of the UP-coded
2 workloads is organized into the one or more concurrency groups at run-time.

1 10. (Original) The method according to claim 1, wherein the MP computer system is a
2 network cache.

1 11. (Currently Amended) A multiprocessor (MP) computer system configured to execute
2 a computer code derived from uniprocessor (UP) coded threads without having to rewrite
3 the UP-coded threads' code, the MP computer system having the computer code, com-
4 prising:

5 a plurality of processors;

6 threads in the uniprocessor coded workloads (UP-workloads) which can execute
7 concurrently, and threads in the UP-workloads which cannot execute concurrently;

8
9 first threads which cannot execute concurrently assigned to a first concurrency
10 group, and second threads which cannot execute concurrently assigned to a second con-
11 currency group, where any thread in the first concurrency group can execute concurrently
12 with any thread in the second concurrency group;

13
14 the UP workload compiled to execute on the MP system, the threads in the first
15 concurrency group compiled to be prevented from executing concurrently, the threads in
16 the second concurrency group compiled to be prevented from executing concurrently, and
17 the threads in the first and the second concurrency groups compiled to execute concur-
18 rently;

19
20 a memory having a plurality of storage locations addressable by the plurality of
21 processors for storing data and program code, the memory being configured to store a
22 separate concurrency-group data structure for each of the first and second a plurality of
23 concurrency groups, each concurrency-group data structure comprising: having,

24 an active-queue pointer storing a location in the memory of an active queue of
25 UP-coded thread messages associated with UP-coded threads in an executable
26 state; and
27 a current-queue pointer storing a location in the memory of a current queue of UP-coded
28 thread messages associated with UP-coded threads waiting to be transferred to the active
29 queue.

1 12. (Original) The MP computer system according to claim 11, wherein each concur-
2 rency-group data structure further comprises a CG flag that stores a value indicating an
3 operational state of a concurrency group associated with the concurrency-group data
4 structure.

1 13. (Original) The MP computer system according to claim 11, wherein each UP-coded
2 thread message stored in the active queue and current queue stores a location in the
3 memory of a top of a call stack associated with a specific UP-coded thread.

1 14. (Original) The MP computer system according to claim 13, wherein the call stack is
2 accessible through a thread control block (TCB) associated with the specific UP-coded
3 thread, the TCB including a CG pointer for storing a memory location of a concurrency-
4 group data structure.

1 15. (Original) The MP computer system according to claim 11, wherein each concur-
2 rency-group data structure further comprises meta-data information associated with a
3 concurrency group.

1 16. (Original) The MP computer system according to claim 11, wherein the MP computer
2 system is a network cache.

1 17. (Currently Amended) An apparatus for executing uniprocessor (UP) coded workloads
2 in a multiprocessor (MP) computer system without having to rewrite the UP-coded work-
3 loads' code, comprising:

4 means for organizing the UP-coded workloads into one or more concurrency
5 groups, wherein UP-coded workloads in the same concurrency group are not permitted to
6 execute concurrently with one another in the MP computer system;

7
8
9 means for identifying threads in the uniprocessor coded workloads (UP-
10 workloads) which can execute concurrently, and identifying threads in the UP-workloads
11 which cannot execute concurrently;

12
13 means for assigning first threads which cannot execute concurrently to a first con-
14 currency group, and assigning second threads which cannot execute concurrently to a
15 second concurrency group, where any thread in the first concurrency group can execute
16 concurrently with any thread in the second concurrency group;

17
18 ~~means for organizing the UP-coded workloads into one or more concurrency~~
19 ~~groups, wherein UP-coded workloads in the same concurrency group are not permitted to~~
20 ~~execute concurrently with one another in the MP computer system;~~

21
22
23 means for compiling the UP workload to execute on the MP system, the threads in
24 the first concurrency group compiled to be prevented from executing concurrently, the
25 threads in the second concurrency group compiled to be prevented from executing con-
26 currently, and the threads in the first and the second concurrency groups compiled to exe-
27 cute concurrently;

29 means for scheduling first and second execution vehicles that respectively execute
30 on different processors in the MP computer system at substantially the same time;

31
32 means for acquiring the ~~[[a]]~~ first concurrency group by the first execution ve-
33 hicle and the ~~[[a]]~~ second concurrency group by the second execution vehicle; and

34
35 means for executing UP-coded workloads in the first concurrency group through
36 the first execution vehicle at substantially the same time as UP-coded workloads in the
37 second concurrency group are executed through the second execution vehicle.

1 18. (Original) The apparatus according to claim 17, wherein the UP-coded workloads are
2 UP-coded threads, and the first and second execution vehicles are first and second pro-
3 cesses.

1 19. (Original) The apparatus according to claim 17, wherein the UP-coded workloads are
2 messages, and the first and second execution vehicles are first and second threads.

1 20. (Original) The apparatus according to claim 17, further comprising:
2 means for dequeuing from a concurrency-group run queue a first concurrency-
3 group data structure associated with the first concurrency group; and
4 means for dequeuing from the concurrency-group run queue a second concur-
5 rency-group data structure associated with the second concurrency group.

1 21. (Original) The apparatus according to claim 20, further comprising:
2 means for setting a first CG flag in the first concurrency-group data structure to a
3 value indicating that the first concurrency group is in a running state; and
4 means for setting a second CG flag in the second concurrency-group data struc-
5 ture to a value indicating that the second concurrency group is in a running state.

1 22. (Original) The apparatus according to claim 20, further comprising:
2 means for appending UP-coded workloads enqueued on a first current queue in
3 the first concurrency-group data structure onto a first active queue in the first concur-
4 rency-group data structure; and
5 means for appending UP-coded workloads enqueued on a second current queue in
6 the second concurrency-group data structure onto a second active queue in the second
7 concurrency-group data structure.

1 23. (Original) The apparatus according to claim 22, further comprising:
2 means for dequeuing UP-coded workloads in the first and second concurrency
3 groups from the first and second active queues, respectively; and
4 means for executing the dequeued UP-coded workloads to completion.

1 24. (Original) The apparatus according to claim 21, further comprising:
2 means for setting the value of the first CG flag to a value indicating that the first
3 concurrency group is in a queued state or in a suspended state; and
4 means for re-enqueueing the first concurrency-group data structure onto the con-
5 currency-group run queue.

1 25. (Currently Amended) A computer-readable media comprising instructions for execu-
2 tion in one or more processors for executing uniprocessor (UP) coded workloads in a
3 multiprocessor (MP) computer system without having to rewrite the UP-coded work-
4 loads' code, comprising:
5
6 identifying threads in the uniprocessor coded workloads (UP-workloads) which
7 can execute concurrently, and identifying threads in the UP-workloads which cannot exe-
8 cute concurrently;
9

10 assigning first threads which cannot execute concurrently to a first concurrency
11 group, and assigning second threads which cannot execute concurrently to a second con-
12 currency group, where any thread in the first concurrency group can execute concurrently
13 with any thread in the second concurrency group;

14
15 organizing the UP-coded workloads into one or more concurrency groups;
16 wherein UP-coded workloads in the same concurrency group are not permitted to execute
17 concurrently with one another in the MP computer system;

18
19
20 compiling the UP workload to execute on the MP system, the threads in the first
21 concurrency group compiled to be prevented from executing concurrently, the threads in
22 the second concurrency group compiled to be prevented from executing concurrently, and
23 the threads in the first and the second concurrency groups compiled to execute concur-
24 rently;

25
26 scheduling first and second execution vehicles that respectively execute on differ-
27 ent processors in the MP computer system at substantially the same time;

28
29 acquiring the [[a]] first concurrency group by the first execution vehicle and
30 the [[a]] second concurrency group by the second execution vehicle; and

31
32 executing UP-coded workloads in the first concurrency group through the first
33 execution vehicle at substantially the same time as UP-coded workloads in the second
34 concurrency group are executed through the second execution vehicle.

1 26. (Original) The computer-readable media according to claim 25, wherein the UP-
2 coded workloads are UP-coded threads, and the first and second execution vehicles are
3 first and second processes.

27. (Original) The computer-readable media according to claim 25, wherein the UP-coded workloads are messages, and the first and second execution vehicles are first and second threads.

28. (Currently Amended) A method for executing workloads in a multiprocessor (MP) computer system, comprising:

identifying threads in an uniprocessor coded workloads (UP-workloads) which can execute concurrently, and identifying threads in the UP-workloads which cannot execute concurrently;

assigning first threads which cannot execute concurrently to a first concurrency group, and assigning second threads which cannot execute concurrently to a second concurrency group, where any thread in the first concurrency group can execute concurrently with any thread in the second concurrency group;

compiling the UP workload to execute on the MP system, the threads in the first concurrency group compiled to be prevented from executing concurrently, the threads in the second concurrency group compiled to be prevented from executing concurrently, and the threads in the first and the second concurrency groups compiled to execute concurrently;

organizing the workloads into one or more concurrency groups, wherein workloads in the same concurrency group are not permitted to execute concurrently with one another in the MP computer system;

scheduling first and second execution vehicles that respectively execute on different processors in the MP computer system at substantially the same time;

acquiring the [[a]] first concurrency group by the first execution vehicle and the [[a]] second concurrency group by the second execution vehicle; and

26 executing workloads in the first concurrency group through the first execution ve-
27 hicle at substantially the same time as workloads in the second concurrency group are
28 executed through the second execution vehicle.

1 29. (Original) The method according to claim 28, wherein the step of acquiring the first
2 and second concurrency groups further comprises:

3 dequeuing from a concurrency-group run queue a first concurrency-group data
4 structure associated with the first concurrency group; and

5 dequeuing from the concurrency-group run queue a second concurrency-group
6 data structure associated with the second concurrency group.

1 30. (Original) The method according to claim 29, further comprising:

2 setting a first CG flag in the first concurrency-group data structure to a value indi-
3 cating that the first concurrency group is in a running state; and

4 setting a second CG flag in the second concurrency-group data structure to a
5 value indicating that the second concurrency group is in a running state.

1 31. (Original) The method according to claim 29, further comprising:

2 appending workloads enqueued on a first current queue in the first concurrency-
3 group data structure onto a first active queue in the first concurrency-group data struc-
4 ture; and

5 appending workloads enqueued on a second current queue in the second concur-
6 rency-group data structure onto a second active queue in the second concurrency-group
7 data structure.

1 32. (Original) The method according to claim 31, further comprising:

2 dequeuing workloads in the first and second concurrency groups from the first
3 and second active queues, respectively; and

4 executing the dequeued workloads to completion.

1 33. (Original) The method according to claim 30, further comprising:

2 in response to the first execution vehicle finishing execution of the workloads in
3 the first concurrency group, the first execution vehicle performing the steps:

4 A) if at least one workload in the first concurrency group is executable:

5 (i) setting the value of the first CG flag to a value indicat-
6 ing that the first concurrency group is in a queued state;

7 (ii) re-enqueueing the first concurrency-group data struc-
8 ture onto the concurrency-group run queue;

9 B) if there are not any workloads in the first concurrency group that are
10 executable, setting the first CG flag to a value indicating that the first concurrency
11 group is in a suspended state;

12 C) dequeuing from the concurrency-group run queue a third concur-
13 rency-group data structure associated with a third concurrency group; and

14 D) setting a third CG flag in the third concurrency-group data structure to
15 a value indicating that the third concurrency group is in a running state.

1 34. (Currently Amended) A method, comprising:

2
3 identifying threads in a uniprocessor coded workload (UP-workload) which can
4 execute concurrently, and identifying threads in the UP-workload which cannot execute
5 concurrently;

6
7 assigning first threads which cannot execute concurrently to a first concurrency
8 group, and assigning second threads which cannot execute concurrently to a second con-
9 currency group, where any thread in the first concurrency group can execute concurrently
10 with any thread in the second concurrency group, and repeating assigning threads which
11 cannot execute concurrently to further concurrency groups to form a plurality of concur-
12 rency groups, where any thread in a concurrency group can execute concurrently with
13 any thread in a different concurrency group;

14
15 compiling the UP workload to execute on the MP system, the threads in the first
16 concurrency group compiled to be prevented from executing concurrently, the threads in
17 the second concurrency group compiled to be prevented from executing concurrently, and
18 the threads in the first and the second concurrency groups compiled to execute concur-
19 rently, and repeating to prevent threads in any of the plurality of concurrency groups
20 from executing concurrently, and permitting threads in different concurrency groups to
21 execute concurrently;

22
23 ~~organizing a plurality of workloads into a plurality of concurrency groups;~~
24 ~~wherein each workload in the same concurrency group are not permitted to execute con-~~
25 ~~currently with another workload in a microprocessor (MP) computer system;~~

26
27 scheduling a plurality of execution vehicles that respectively execute on different
28 processors in the MP computer system at substantially the same time;

29 acquiring by each execution vehicle of the plurality of execution vehicles a con-
30 currency group from the plurality of concurrency groups; and

31 executing workloads in the plurality of concurrency groups through the plurality
32 of execution vehicles at substantially the same time.

1 35. (Previously Presented) The method according to claim 34, wherein the workloads are
2 uniprocessor (UP) coded threads, and the plurality of vehicles are processes.

1 36. (Previously Presented) The method according to claim 34, wherein the workloads are
2 messages, and the plurality of vehicles are first and second threads.

1 37. (Previously Presented) The method according to claim 34, wherein the workloads
2 are uniprocessor (UP) coded workloads.